Capstone Project 2 Proposal

Home Credit Default Risk

Overview :

There are lots of people who do not particularly have a prior credit history, for example students, small businessmen, etc. who need credits, be it for studies, or for setting up some sort of businesses. Without adequate credit history, the lending organizations find it difficult to lend credits to such people, as these loans could be associated with high risks. In these kinds of situations, some lending organizations even tend to exploit the borrowers by asking for too high of an interest rate.

There are another subset of people, who do have prior credit history, which could be with the same organization or some other organizations. However, going through that historical data could be very time consuming and redundant. This would scale up even further as the number of applicants increases.

For such cases, if there could be a way through which the lending organization could predict or estimate the borrower's repayment capability, the process could be streamlined and be made effective for both the lender and the borrower. It could save resources both in terms of humans and time.

So the main two questions that the lender needs answer to are:

1) How risky is the borrower?

2) Given the borrower's risk, should we give him/her loan?

Business Objective: Home Credit Default Risk

Home Credit strives to broaden financial inclusion for the unbanked population by providing a positive and safe borrowing experience. In order to make sure this underserved population has a positive loan experience, Home Credit makes use of a variety of alternative data--including telco and transactional information--to predict their clients' repayment abilities.

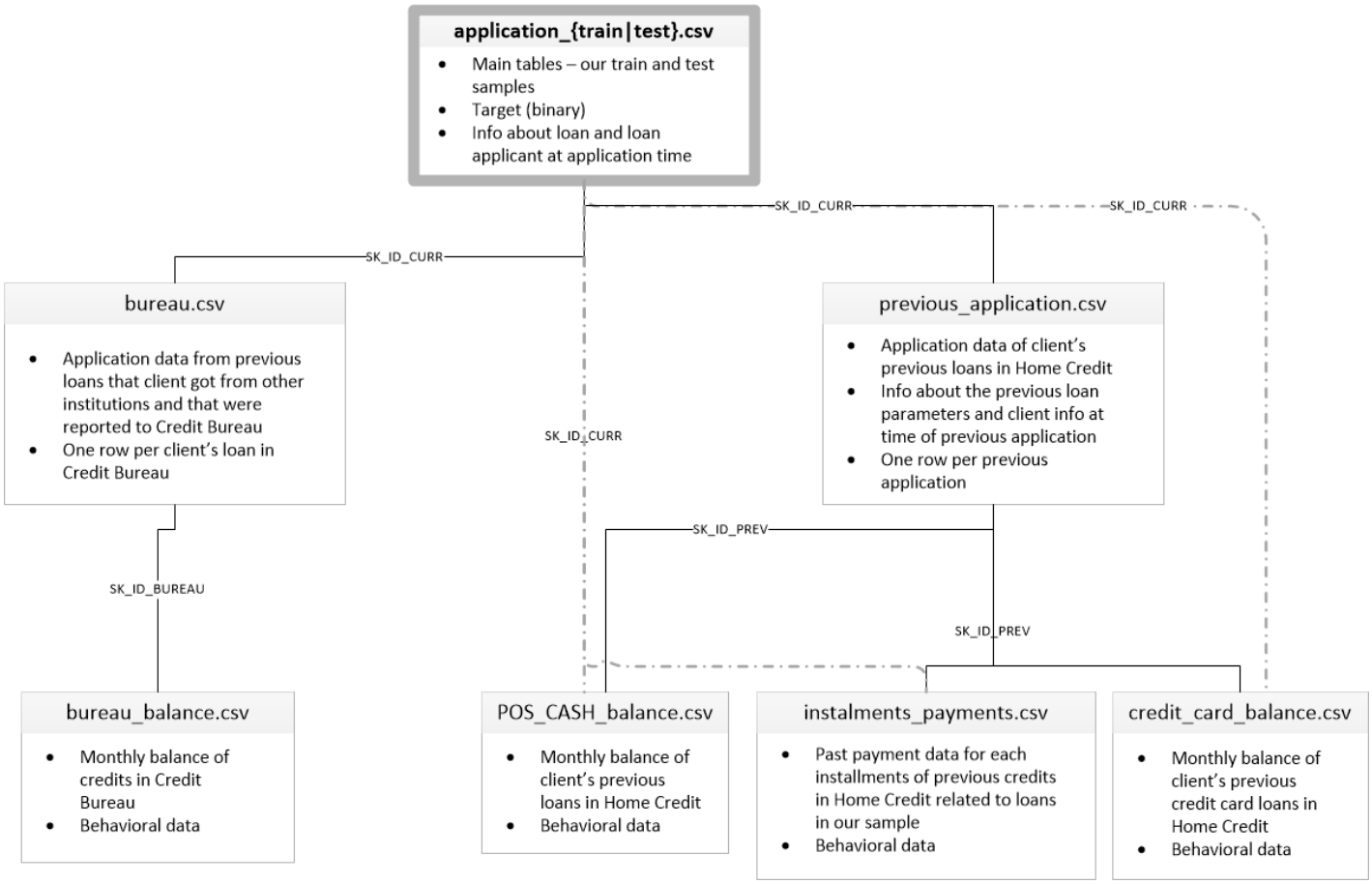
Home depot wants to have a model that clients capable of repayment are not rejected and that loans are given with a principal, maturity, and repayment calendar that will empower their clients to be successful.

Data:

- application_{train|test}.csv
    - This is the main table, broken into two files for Train (with TARGET) and Test (without TARGET).
    - Static data for all applications. One row represents one loan in our data sample.
- bureau.csv
    - All client's previous credits provided by other financial institutions that were reported to Credit Bureau (for clients who have a loan in our sample).
    - For every loan in our sample, there are as many rows as number of credits the client had in Credit Bureau before the application date.
- bureau_balance.csv
    - Monthly balances of previous credits in Credit Bureau.
    - This table has one row for each month of history of every previous credit reported to Credit Bureau – i.e the table has (#loans in sample * # of relative previous credits * # of months where we have some history observable for the previous credits) rows.
- POS_CASH_balance.csv
    - Monthly balance snapshots of previous POS (point of sales) and cash loans that the applicant had with Home Credit.
    - This table has one row for each month of history of every previous credit in Home Credit (consumer credit and cash loans) related to loans in our sample – i.e. the table has (#loans in sample * # of relative previous credits * # of months in which we have some history observable for the previous credits) rows.
- credit_card_balance.csv
    - Monthly balance snapshots of previous credit cards that the applicant has with Home Credit.
    - This table has one row for each month of history of every previous credit in Home Credit (consumer credit and cash loans) related to loans in our sample – i.e. the table has (#loans in sample * # of relative previous

credit cards * # of months where we have some history observable for the previous credit card) rows.

- previous_application.csv
    - All previous applications for Home Credit loans of clients who have loans in our sample.
    - There is one row for each previous application related to loans in our data sample.
- installments_payments.csv
    - Repayment history for the previously disbursed credits in Home Credit related to the loans in our sample.
    - There is a) one row for every payment that was made plus b) one row each for missed payment.
    - One row is equivalent to one payment of one installment OR one installment corresponding to one payment of one previous Home Credit credit related to loans in our sample.
- HomeCredit_columns_description.csv
    - This file contains descriptions for the columns in the various data files.



WorkFlow:

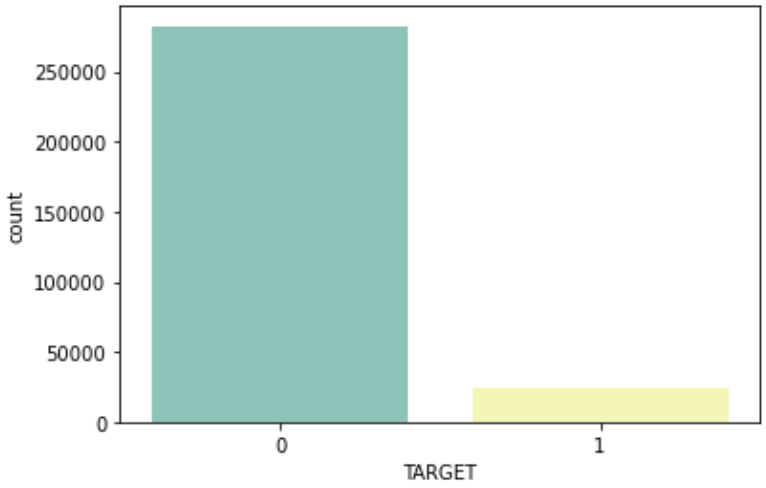
- Collecting data and applying data wrangling
- Exploring the data and do some data analysis to find trends and story telling
- Find the relationship between different variables
- Feature selection and PCA
- Implement Machine Learning algorithms
- Model evaluation and validation

Deliverables:

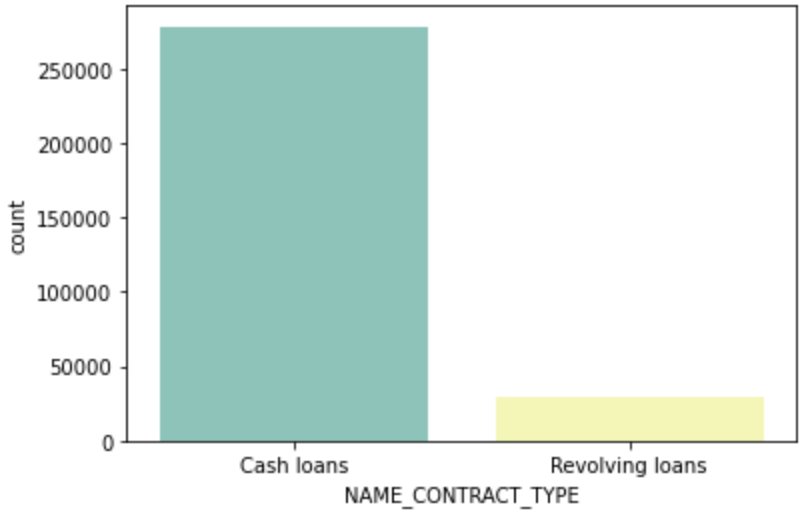
- Jupyter notebook (.ipynb code)
- Final report
- Slide deck
- Presentation

Implementation:

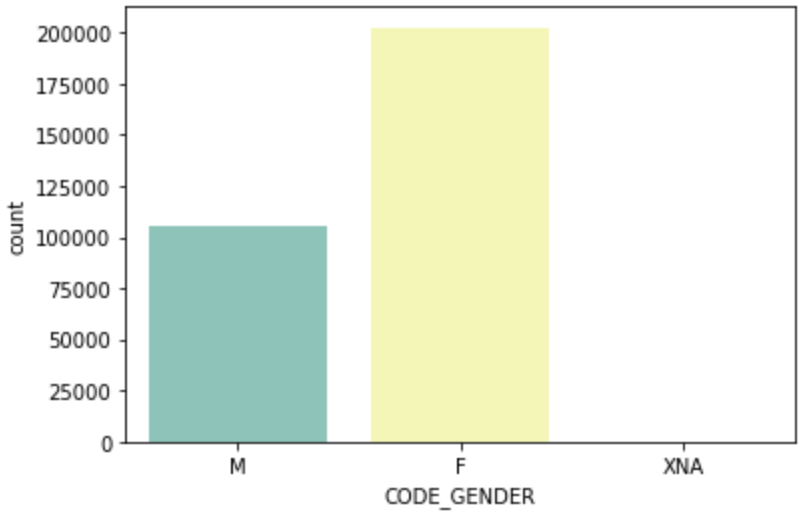
Exploratory Data Analysis:



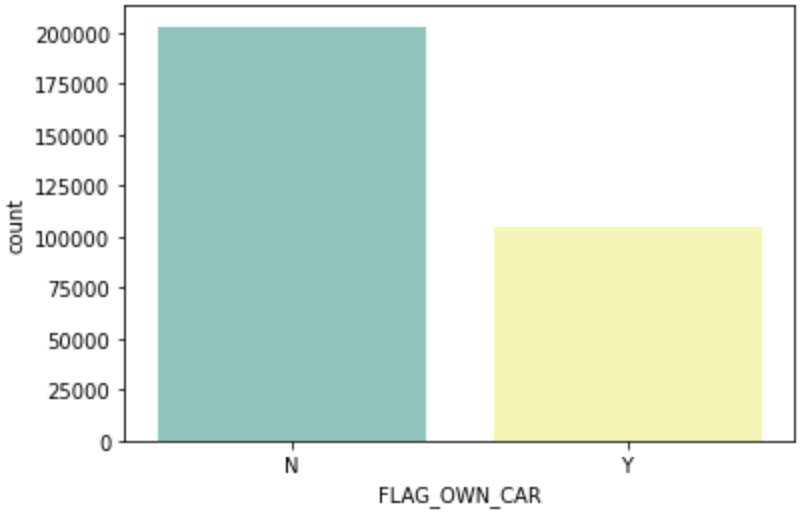
Target is very unbalaced which is normal. Which means most peopel are cleared to get a loan



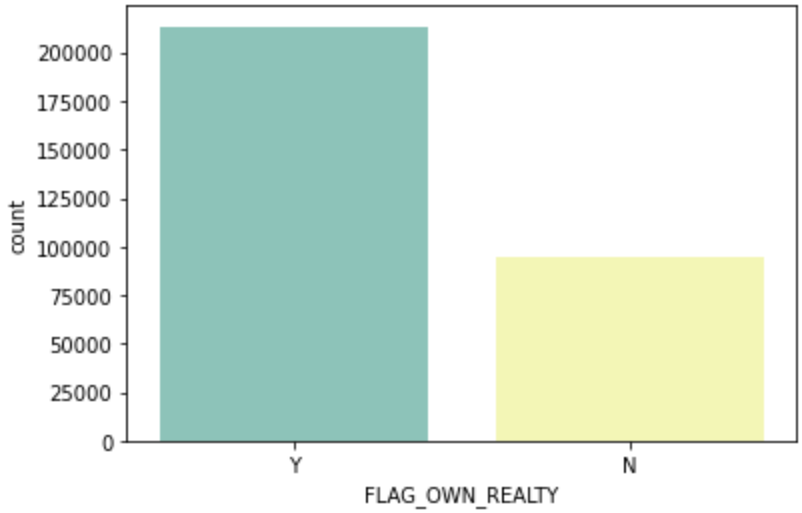
Around 90% of contract type is cash loan and 10% is revolving loan



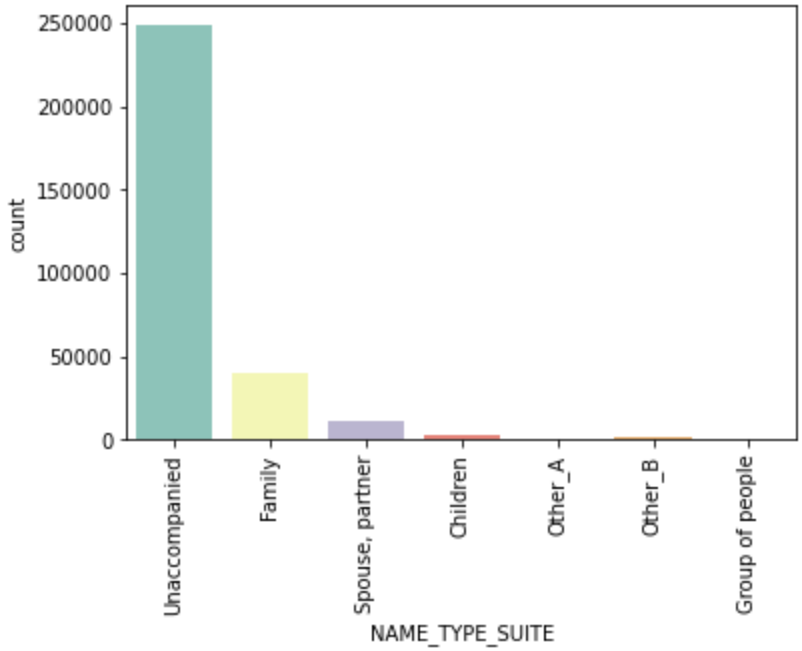
34% are male in the applicants.



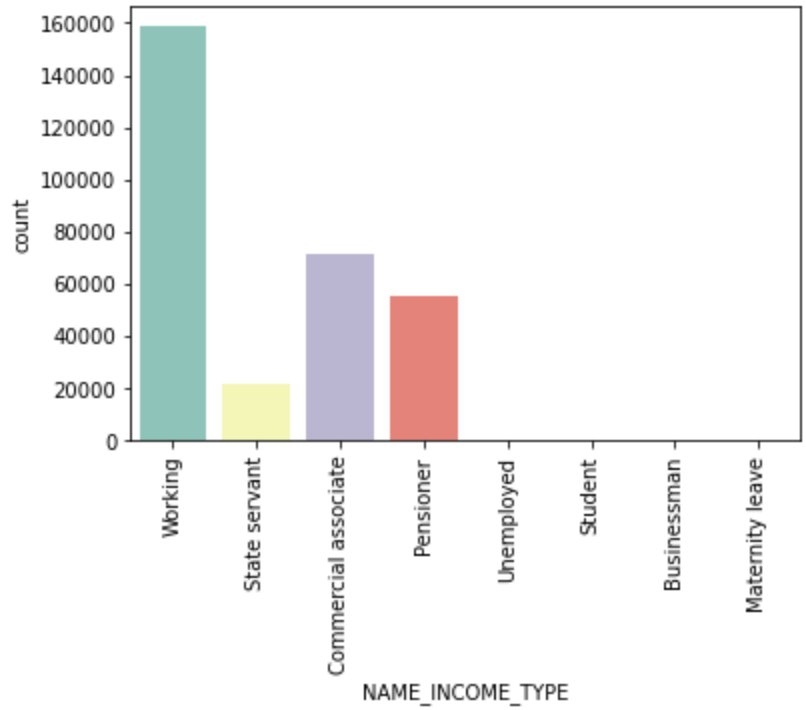
About 66% don’t have a car



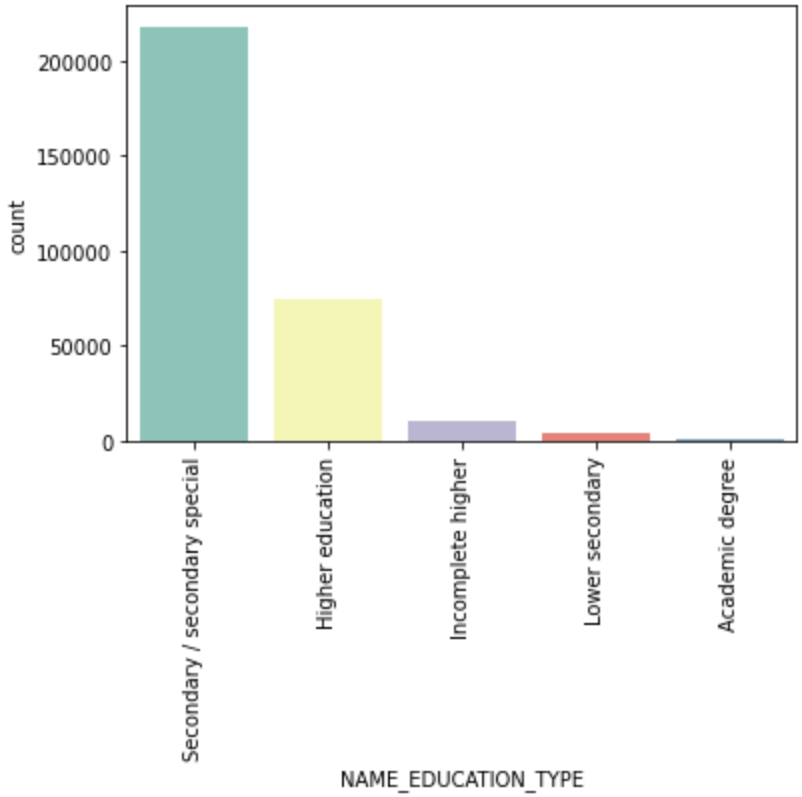
About 70% have their realty



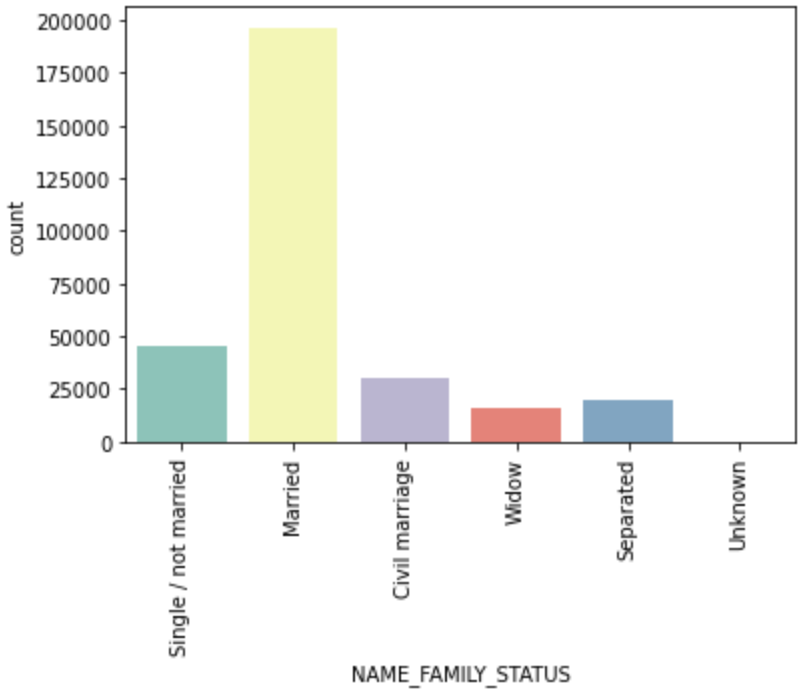
Most applicants are unaccompanied



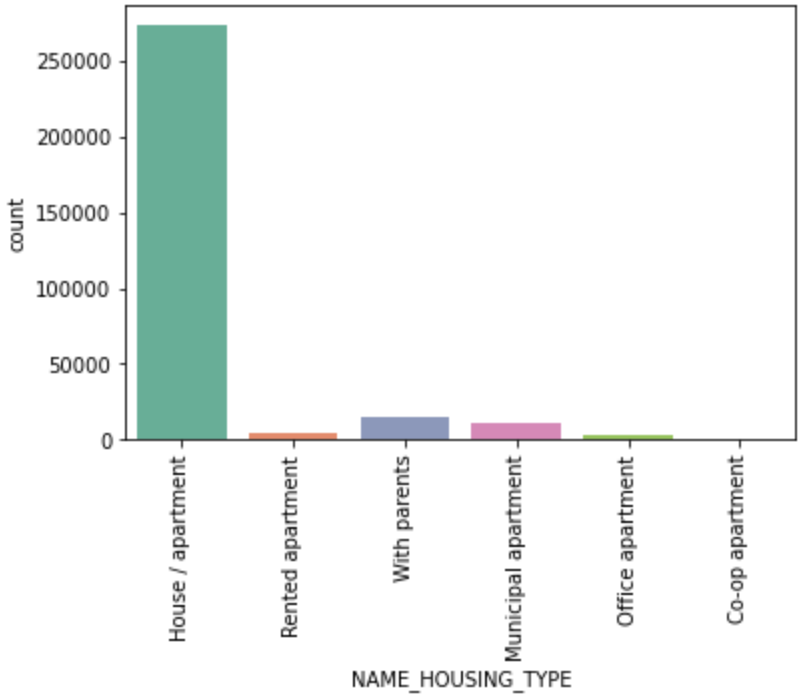
Income type of most applicant is from working



Education type is mostly secondary



Family Satuts is mostly married



House type is mostly house/apartment

Missing values:

	Missing Values	% of Total Values
COMMONAREA_MEDI	214865	69.9
COMMONAREA_AVG	214865	69.9
COMMONAREA_MODE	214865	69.9
NONLIVINGAPARTMENTS_MEDI	213514	69.4
NONLIVINGAPARTMENTS_MODE	213514	69.4
NONLIVINGAPARTMENTS_AVG	213514	69.4
FONDKAPREMONT_MODE	210295	68.4
LIVINGAPARTMENTS_MODE	210199	68.4
LIVINGAPARTMENTS_MEDI	210199	68.4
LIVINGAPARTMENTS_AVG	210199	68.4
FLOORSMIN_MODE	208642	67.8
FLOORSMIN_MEDI	208642	67.8
FLOORSMIN_AVG	208642	67.8
YEARS_BUILD_MODE	204488	66.5
YEARS_BUILD_MEDI	204488	66.5
YEARS_BUILD_AVG	204488	66.5
OWN_CAR_AGE	202929	66.0
LANDAREA_AVG	182590	59.4
LANDAREA_MEDI	182590	59.4
LANDAREA_MODE	182590	59.4

When it comes time to build our machine learning models, we will have to fill in these missing values (known as imputation). In later work, we will use models such as XGBoost that can handle missing values with no need for imputation. Another option would be to drop columns with a high percentage of missing values, although it is impossible to know ahead of time if these columns will be helpful to our model. Therefore, we will keep all of the columns for now.

Column type:

We need to deal with categorical variables(objects are categorical). We can look more to see how many different categories each obj has

```
NAME_CONTRACT_TYPE            2
CODE_GENDER                   3
FLAG_OWN_CAR                  2
FLAG_OWN_REALTY               2
NAME_TYPE_SUITE               7
NAME_INCOME_TYPE              8
NAME_EDUCATION_TYPE           5
NAME_FAMILY_STATUS            6
NAME_HOUSING_TYPE             6
OCCUPATION_TYPE              18
WEEKDAY_APPR_PROCESS_START    7
ORGANIZATION_TYPE            58
FONDKAPREMONT_MODE            4
HOUSETYPE_MODE                3
WALLSMATERIAL_MODE            7
EMERGENCYSTATE_MODE           2
dtype: int64
```

We can see for the most part the categorical features are not too many

Changing categorical features:

I used one ot encoding to do so.

Aligning Training and Testing Data

There need to be the same features (columns) in both the training and testing data. One-hot encoding has created more columns in the training data because there were some categorical variables with categories not represented in the testing data. To remove the columns in the training data that are not in the testing data, we need to align the dataframes. First we extract the target column from the training data (because this is not in the testing data but we need to keep this information). When we do the align, we must make sure to set axis = 1 to align the dataframes based on the columns and not on the rows!

Classification:

This is a supervised classification task:

- Supervised: The labels are included in the training data and the goal is to train a model to learn to predict the labels from the features
- Classification: The label is a binary variable, 0 (will repay loan on time), 1 (will have difficulty repaying loan)

Different classification techniques have been implemented for this project. I split the data into training and testing and also used cross validation for this.

Evaluation Metirc:

For evaluation I use ROC AUC

A single line on the graph indicates the curve for a single model, and movement along a line indicates changing the threshold used for classifying a positive instance. The threshold starts at 0 in the upper right to and goes to 1 in the lower left. A curve that is to the left and above another curve indicates a better model. For example, the blue model is better than the red model, which is better than the black diagonal line which indicates a naive random guessing model.

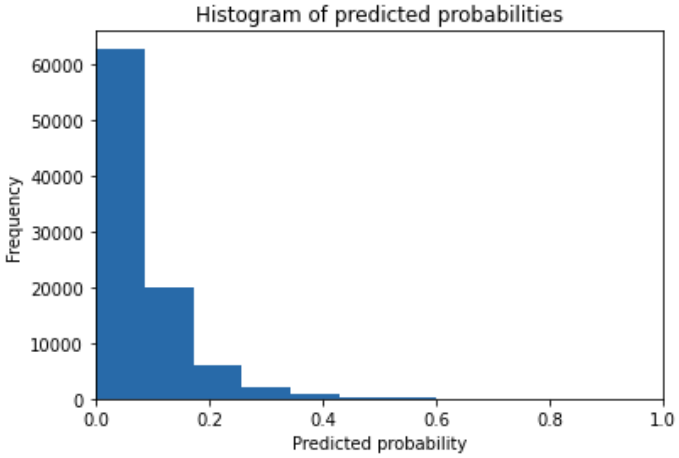
The Area Under the Curve (AUC) explains itself by its name! It is simply the area under the ROC curve. (This is the integral of the curve.) This metric is between 0 and 1 with a better model scoring higher. A model that simply guesses at random will have an ROC AUC of 0.5.

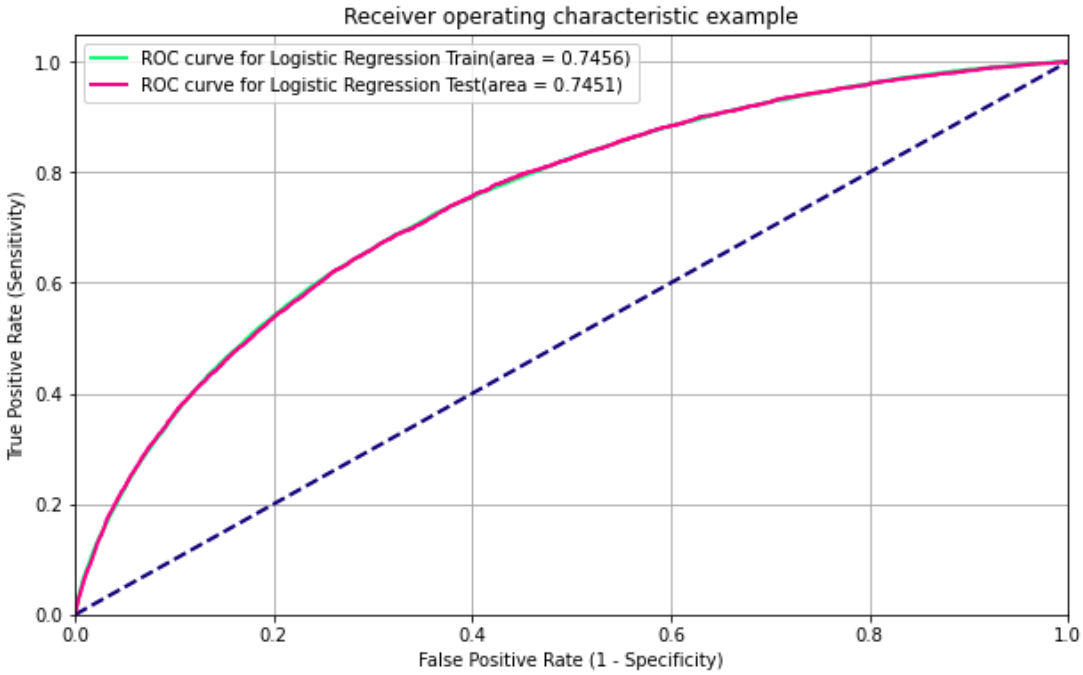
When we measure a classifier according to the ROC AUC, we do not generation 0 or 1 predictions, but rather a probability between 0 and 1. This may be confusing because we usually like to think in terms of accuracy, but when we get into problems with inbalanced classes (we will see this is the case), accuracy is not the best metric. A model with a high ROC AUC will also have a high accuracy, but the ROC AUC is a better representation of model performance.

Logistic Regression:

To get a baseline, we will use all of the features after encoding the categorical variables. We will preprocess the data by filling in the missing values (imputation) and normalizing the range of the features (feature scaling). After that I used Logistic regression as a baseline.

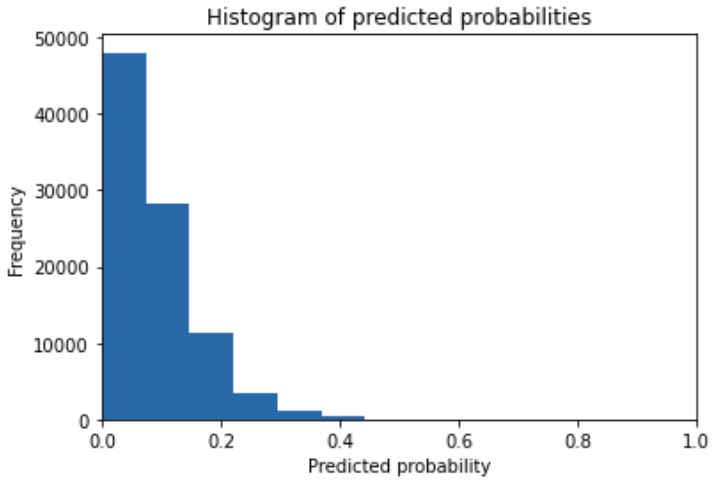
```
Correct accuracy of the train set with Logistic Regression is: 91.8758 %
Correct accuracy of the test set with Logistic Regression is: 92.0209 %

Total elapsed time is: 8.5649  sec

Confusion Matrix is:
[[84817    97]
 [ 7264    76]]

Logistic Regression log_loss is: 2.7559

Average precision-recall score: 0.0833
```

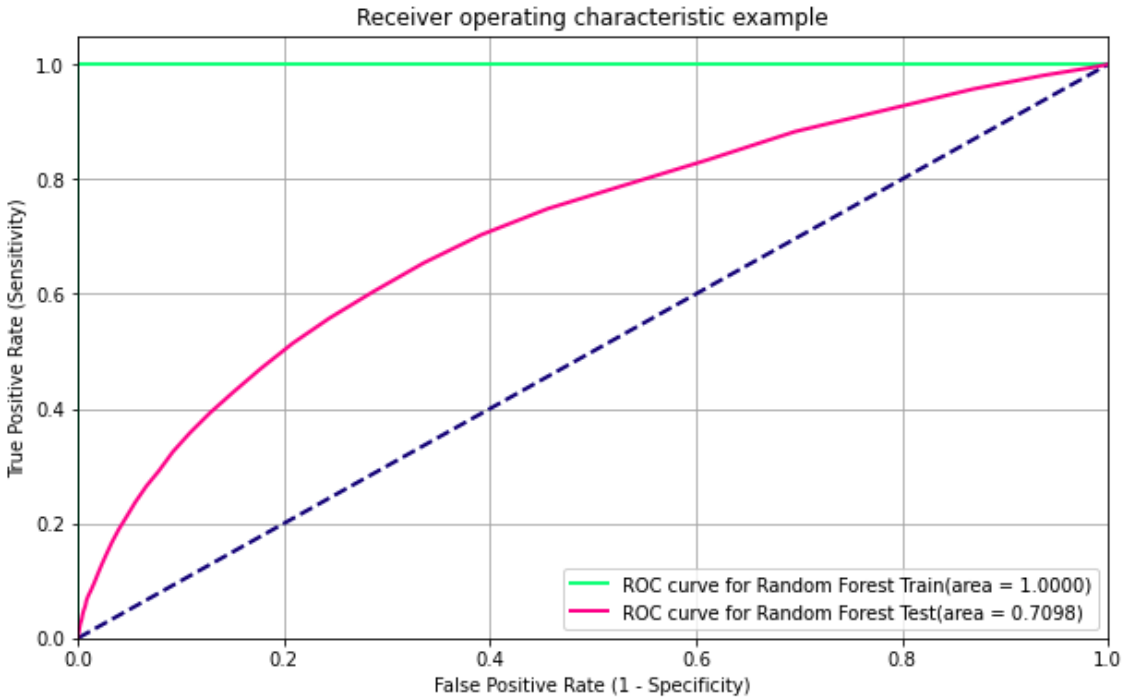




Random Forest:

```
Total elapsed time is: 139.5264  sec
Correct accuracy of the train set with Random Forest Classifier is: 99.9977 %
Correct accuracy of the test set with Random Forest Classifier is: 92.0545 %


Confusion Matrix is:
[[84913     1]
 [ 7329    11]]

Random Forest log_loss is: 2.7443

Average precision-recall score: 0.0808
```





We can see there is overfitting on training data.

I will set hyperparameter to overcome the overfitting: n_estimators = number of trees in the foreset

max_depth = max number of levels in each decision tree

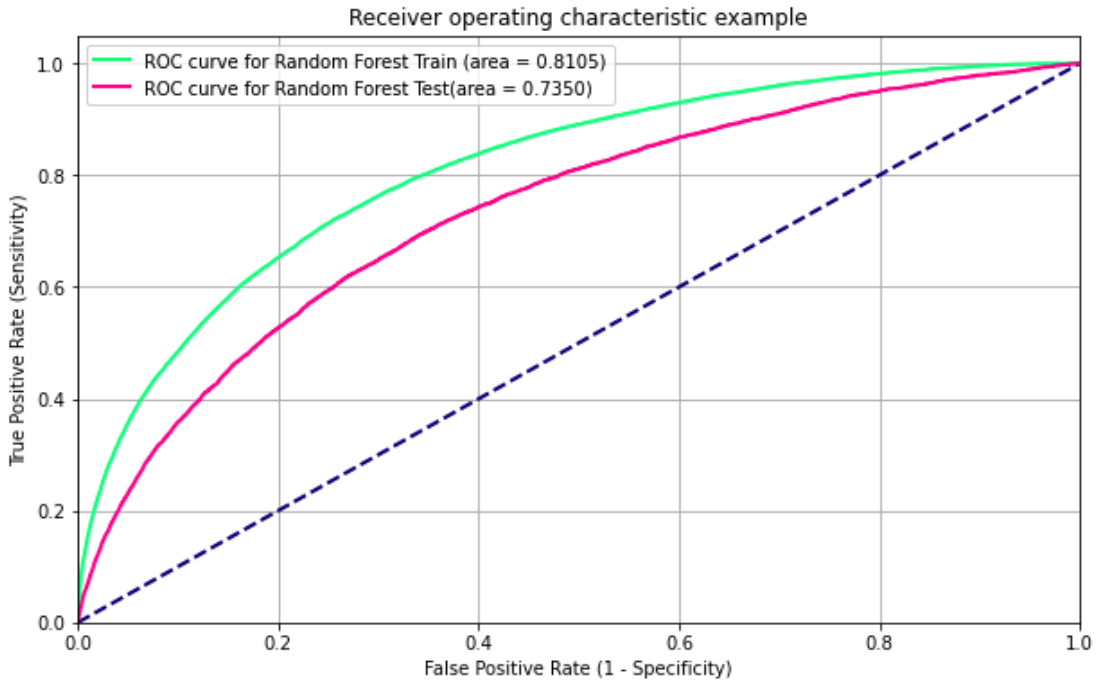
min_samples_split = min number of data points placed in a node before the node is split

min_samples_leaf = min number of data points allowed in a leaf node

bootstrap = method for sampling data points (with or without replacement)

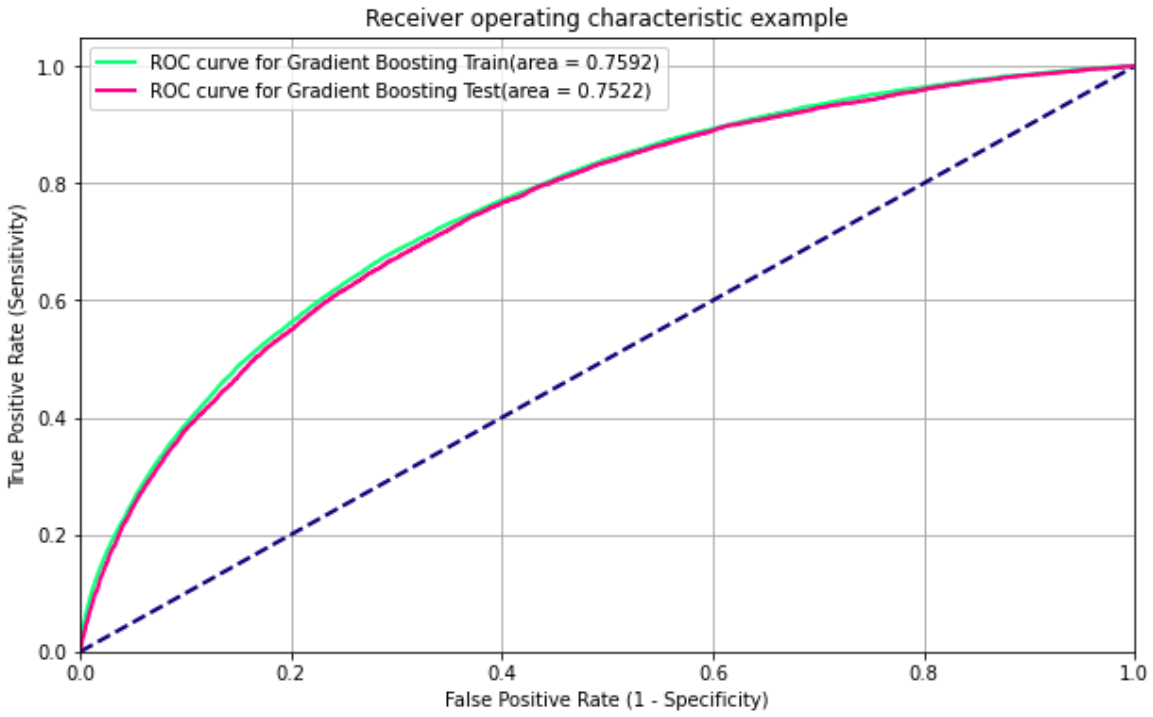
Random Forest with Hyperparameter:

```
Total elapsed time is: 54.3217  sec
Correct accuracy of the train set with Random Forest Classifier is: 91.8776 %
Correct accuracy of the test set with Random Forest Classifier is: 92.0437 %

Confusion Matrix is:
[[84914     0]
 [ 7340     0]]

Random Forest log_loss is: 2.748

Average precision-recall score: 0.0796
```



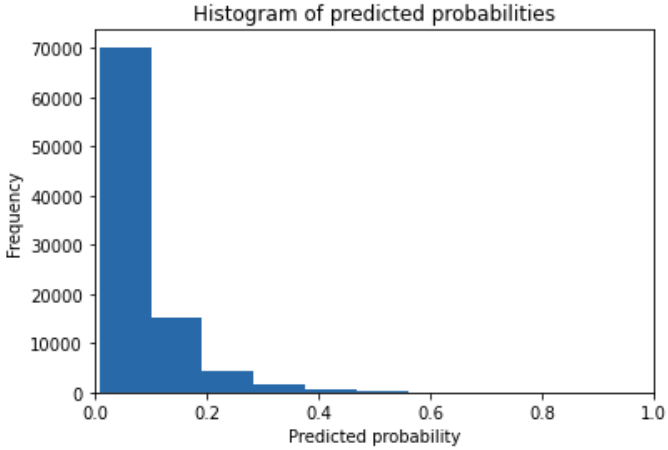
Gradient Boosting:

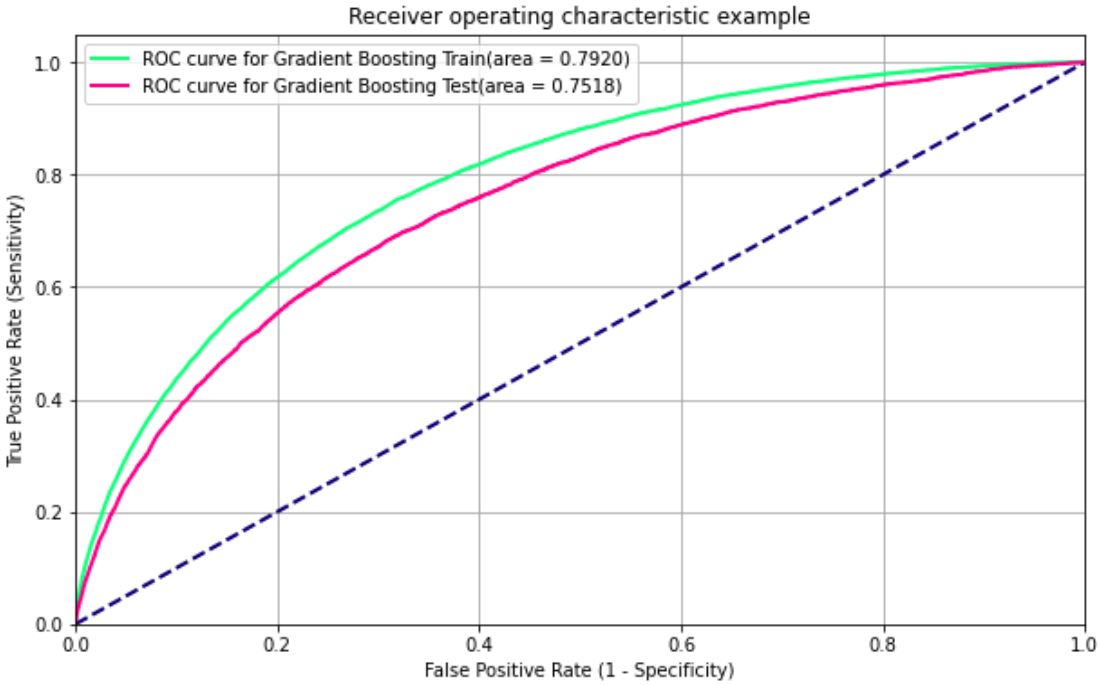
```
Total elapsed time is: 291.6353  sec
Correct accuracy of the train set with Gradient Boosting Classifier is: 91.9533 %
Correct accuracy of the test set with Gradient Boosting Classifier is: 92.0372 %

Confusion Matrix is:
[[84811   103]
 [ 7243    97]]

GradClassifier log_loss is: 2.7503

Average precision-recall score: 0.0849
```



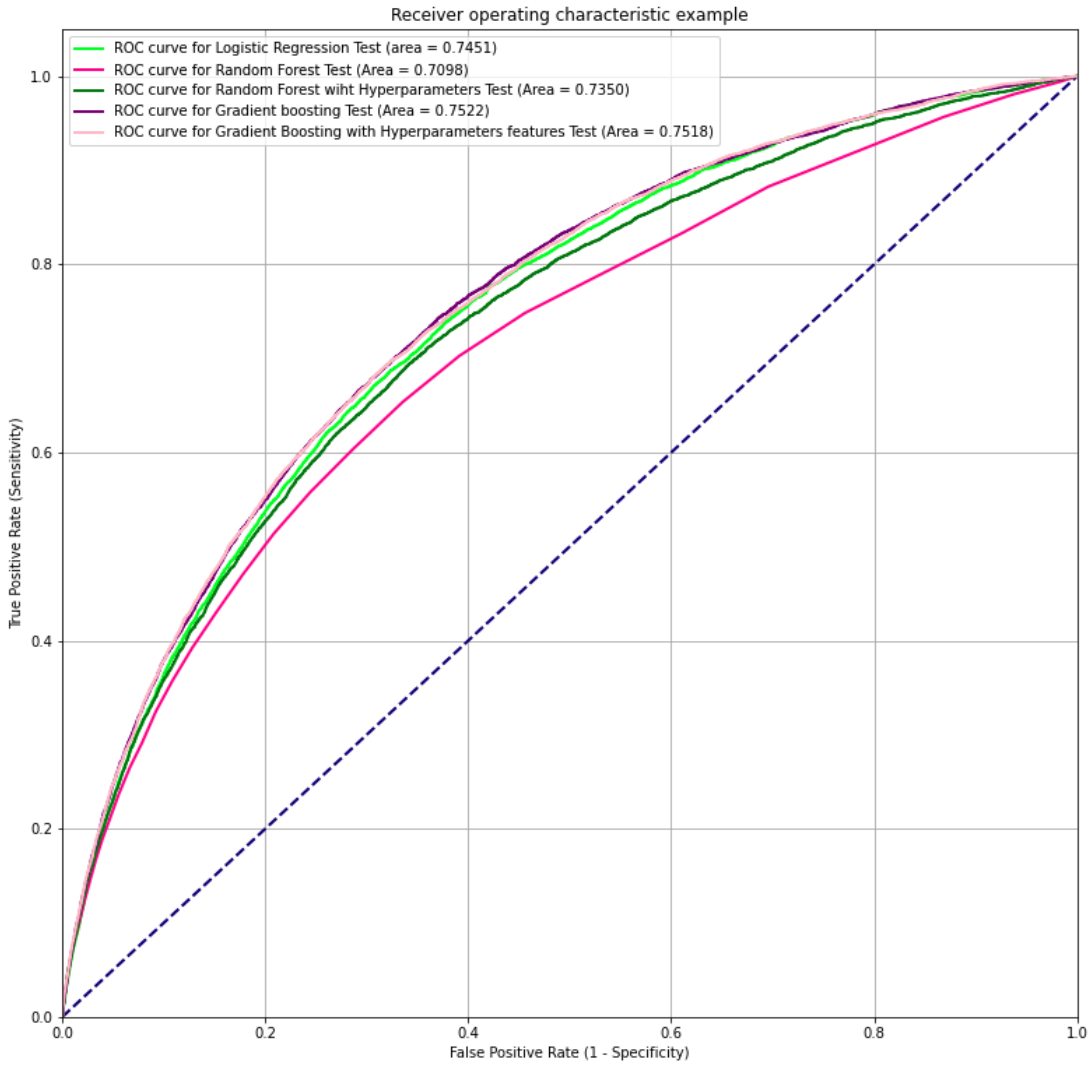
Gradient Boosting with Hyper parameter:

```
Total elapsed time is: 33.9333  sec
Correct accuracy of the train set with Gradient Boosting Classifier is: 91.9691 %
Correct accuracy of the test set with Gradient Boosting Classifier is: 92.073 %

Confusion Matrix is:
[[84824    90]
 [ 7223   117]]

GradClassifier log_loss is: 2.7379

Average precision-recall score: 0.0873
```





Comparing different AUC:



Conclusion and future work:

In this project, I used the kaggle machine learning competition. First I made sure I understand the data and the problem I am trying to solve. For this I performed some EDA to find some trends and relationships in the data. After that I did some preprocessing like encoding categorical features, imputation and scaling.

In this project I only used the main csv file . For future work we can use all of the csv files we have to get a better model.

We can also improve the models by doing more feature engineering in the future.